Figure 1a

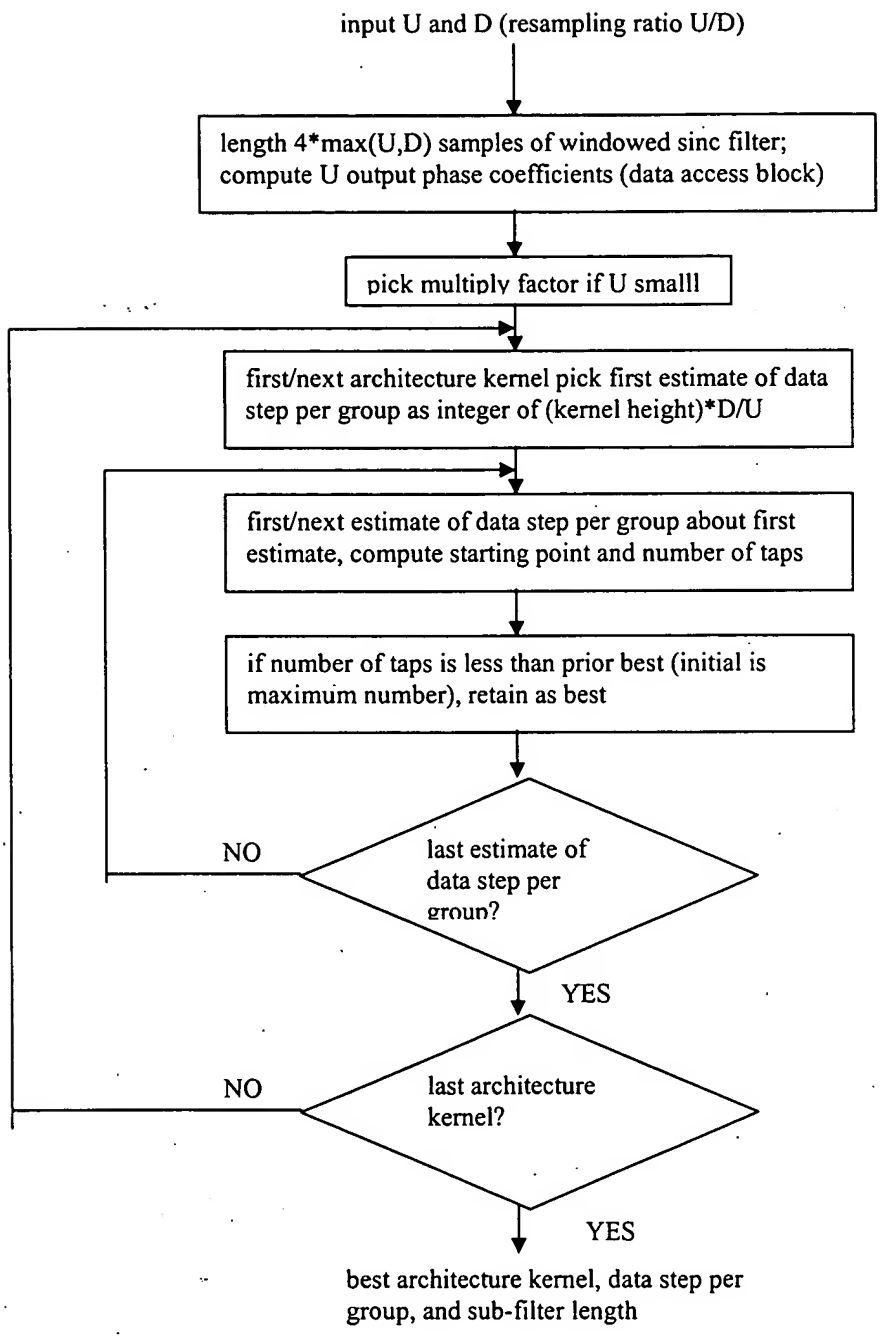

input U and D (resampling ratio U/D)
length 4*max(U,D) samples of windowed sinc filter; compute U output phase coefficients (data access block)
pick multiply factor if U smalll
first/next architecture kernel pick first estimate of data step per group as integer of (kernel height)*D/U
first/next estimate of data step per group about first estimate, compute starting point and number of taps
if number of taps is less than prior best (initial is maximum number), retain as best
NO
last estimate of data step per groun?
YES
NO
last architecture kernel?
YES
best architecture kernel, data step per group, and sub-filter length

Figure 1b (DSC components)

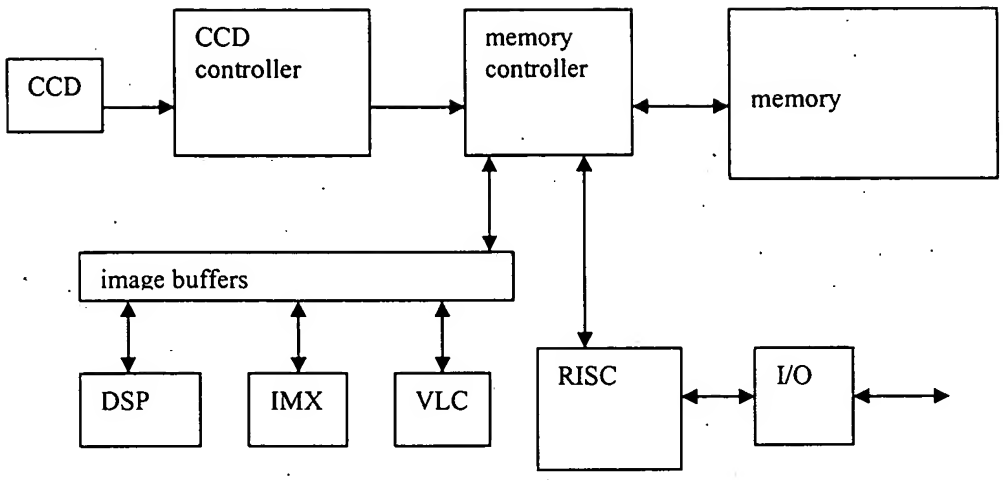


Figure 2a

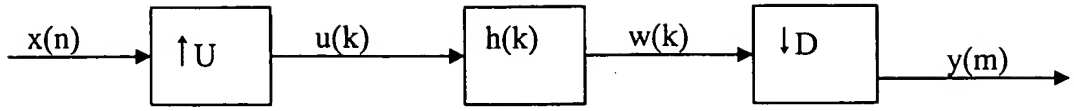


Figure 2b

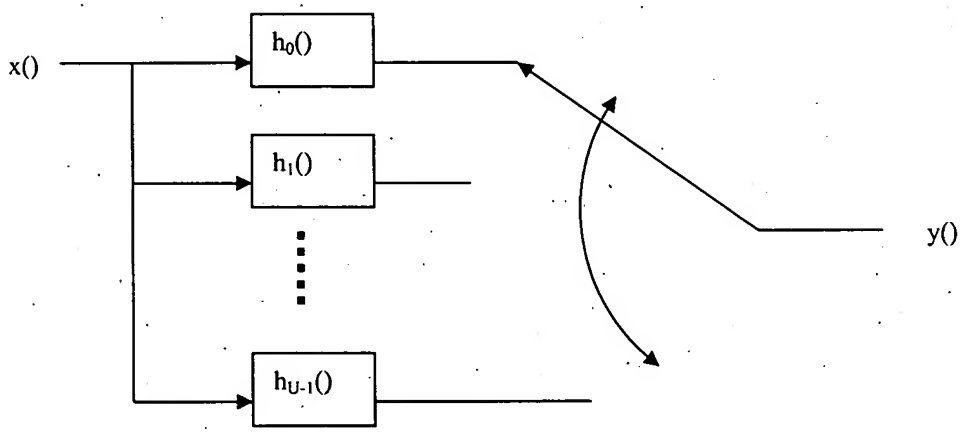
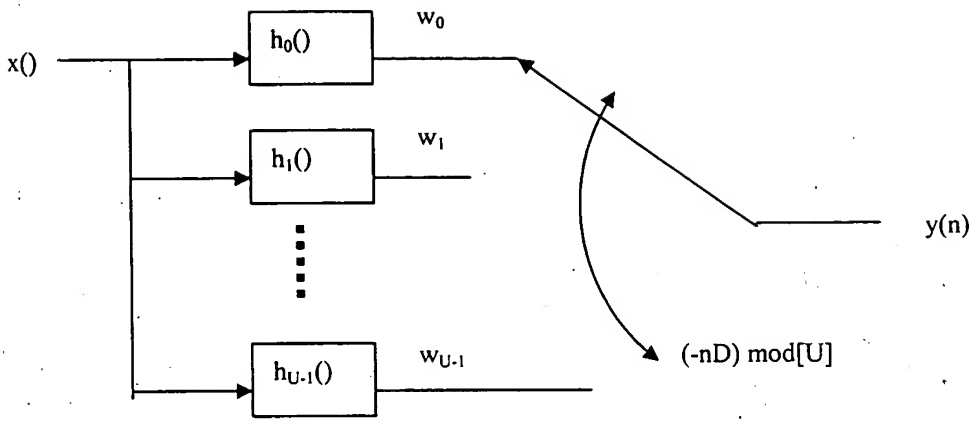


Figure 2c



T 35502

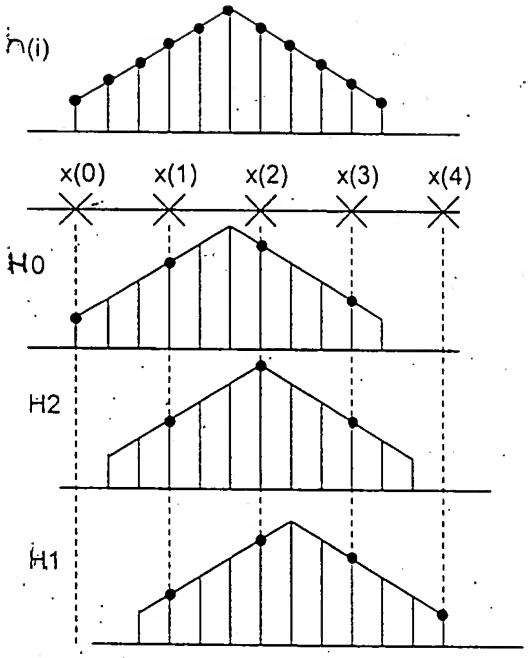


Fig 3a

For polyphase filtering implementation of this upsampling, the data access pattern is shown below.

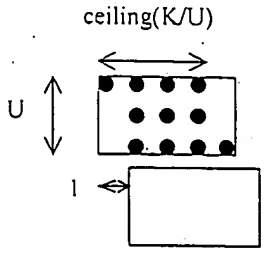


Fig. 3b

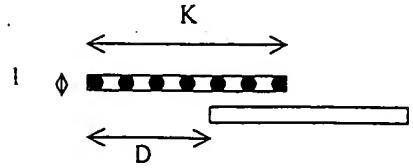


Fig. 3c

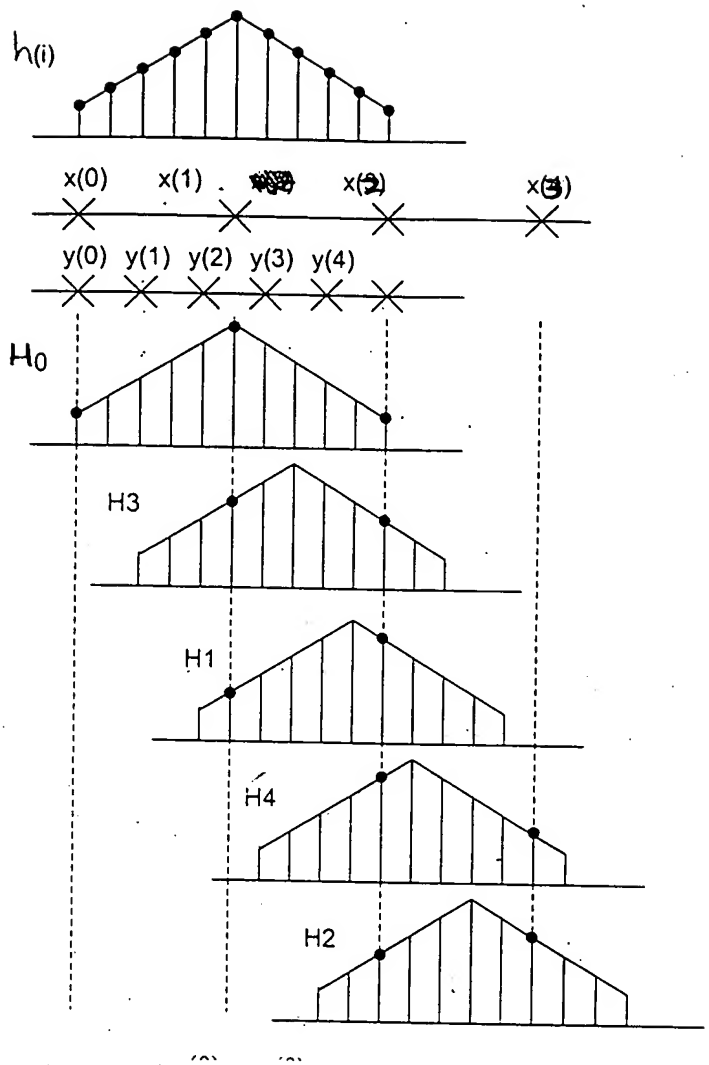


Fig. 4a

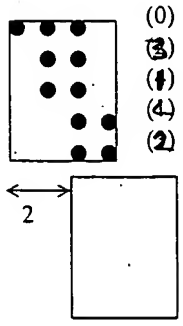


Fig. 4b

T35502

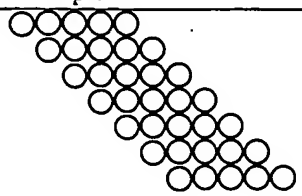
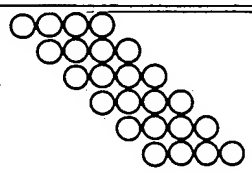
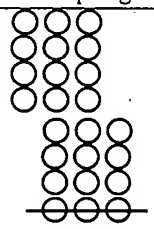
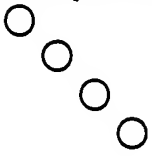
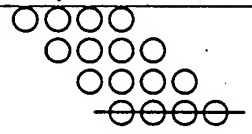
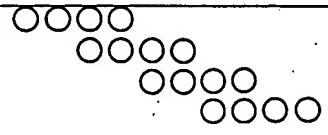
Architecture Kernel	Explanation	Example
single point 	Single-thread filtering, getting one data point at a time. The next access can stay on the same point, or move for a fixed distant. This is the most flexible pattern. However, there is usually time associated with each level of looping, which is needed to implement changing of stepping distance in a regular manner. A parallel DSP performing vertical filtering often use this single-point kernel, as its parallelism is applied to different filtering problems (for vertical filtering, each column is operated independently and is thus a separate problem.)	 This is for 5-tap-per-output filtering going on for 7 outputs, for one data access block for 7/D resampling
4-wide 	DSP with fixed, 4-wide, data access, and capability to compute inner-product with coefficient array and producing a single sum. To be efficient for filtering, the starting point should be on any alignment.	 This is 4-tap-per-output filtering for a 6/D resampling
4-tall 	DSP with 4 parallel execution units, and they are all fed with the same single data point. There is a significant distinction in DSP architecture that affects how we can use this and many other architecture kernels: writing in any alignment, or writing only on 2^N-word alignment. The former can be used to implement any U factor. The latter can only work with U being multiple of 2^N. When U is not a multiple of 4, for example, we upscale U and D so that they are, at the expense of efficiency.	 This is a write-any-alignment architecture implementing a 7/D resampling with 3-tap-per-output filters. Note that 8 outputs are computed and then one is thrown away.
1:1 slope, 4 outputs 	DSP with 4 parallel execution units, and they are fed with 4 data points, one for each. To be efficient for filtering, the 4 inputs need to be on any word alignment.	 This for a 3/D resampling, with 4-tap-per-output filters. Note that 4 outputs are computed and then one is thrown away.
1:2 slope, 2 outputs 	DSP that can take in 4 inputs, and perform acc_A = acc_A + c0*d0 + c1*d1, acc_B = acc_B + c2*d2 + c3*d3 To be efficient for filtering, the inputs need to be on any word alignment.	 This for a 4/D resampling, with 4-tap-per-output filters.

Fig. 5

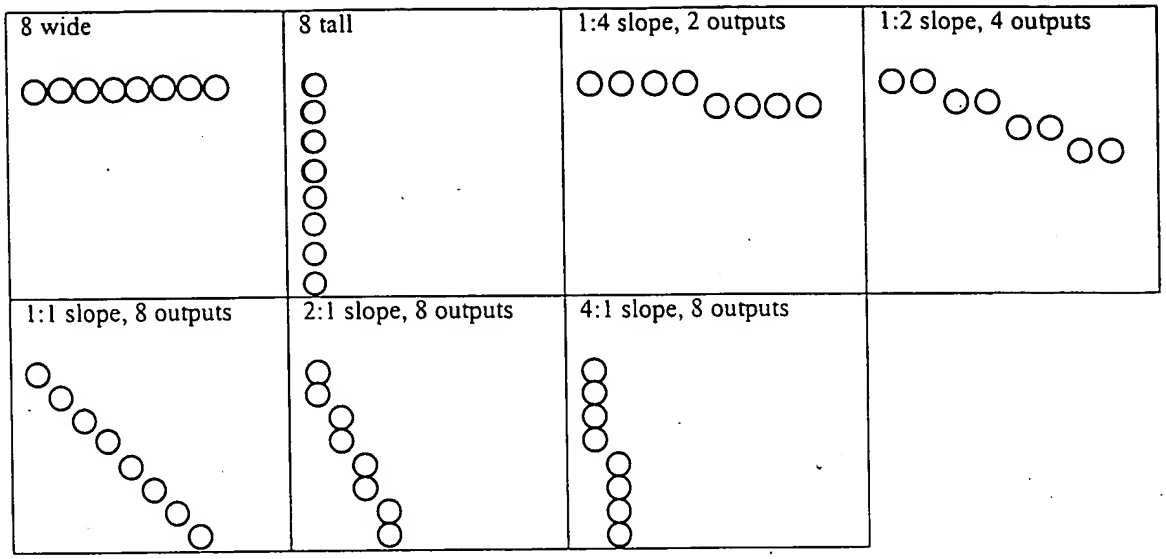

8 wide
8 tall
1:4 slope, 2 outputs
1:2 slope, 4 outputs
1:1 slope, 8 outputs
2:1 slope, 8 outputs
4:1 slope, 8 outputs

Fig. 6a

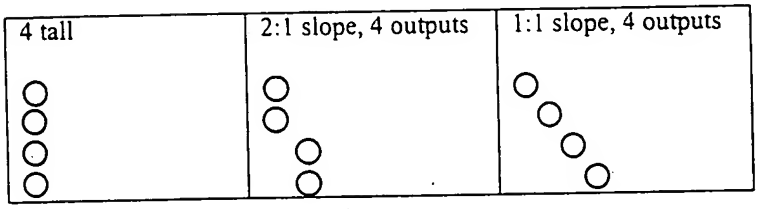

4 tall
2:1 slope, 4 outputs
1:1 slope, 4 outputs

Fig. 6b

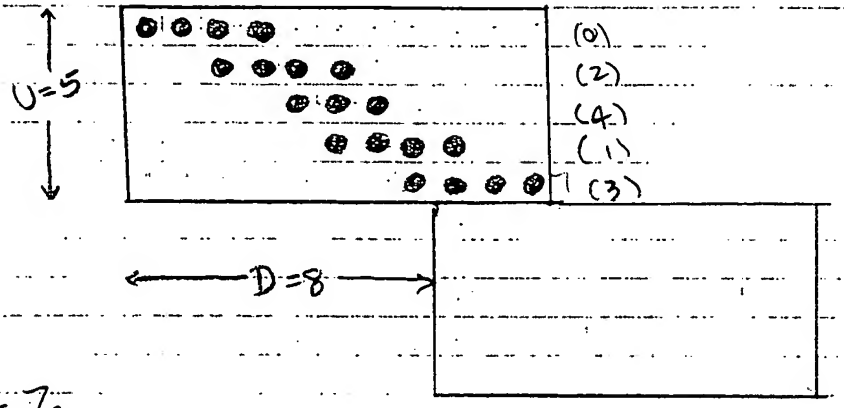


Fig. 7a

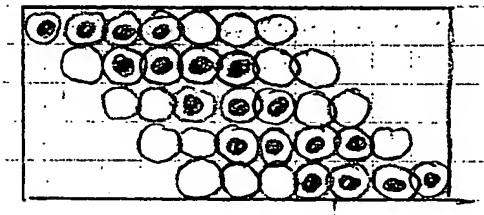


Fig. 7c

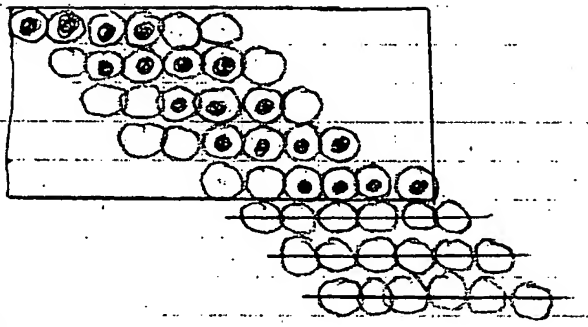


Fig. 7b

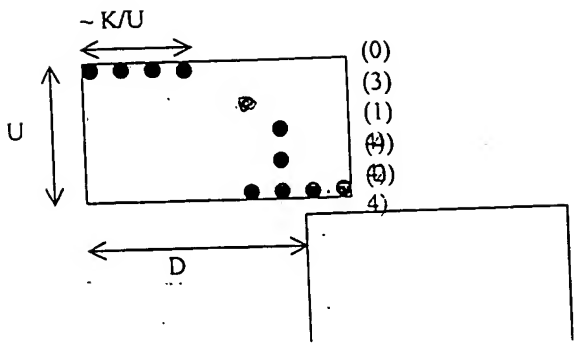
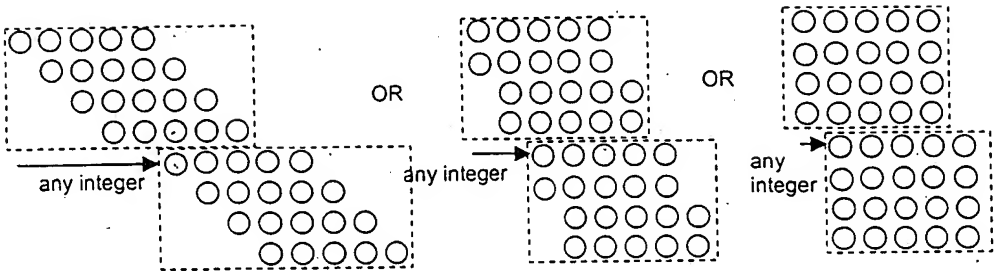


Fig. 8a

Horizontal resampling



Vertical resampling

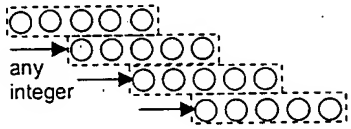


Fig. 8b

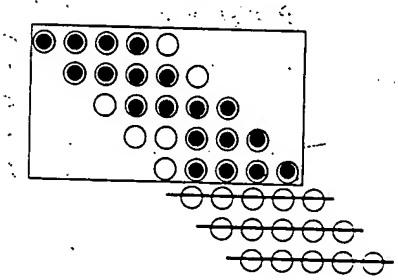


Fig. 8c

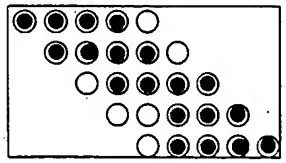


Fig 8d

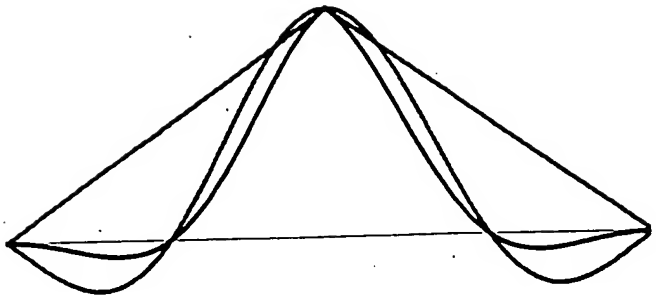


Fig. 9

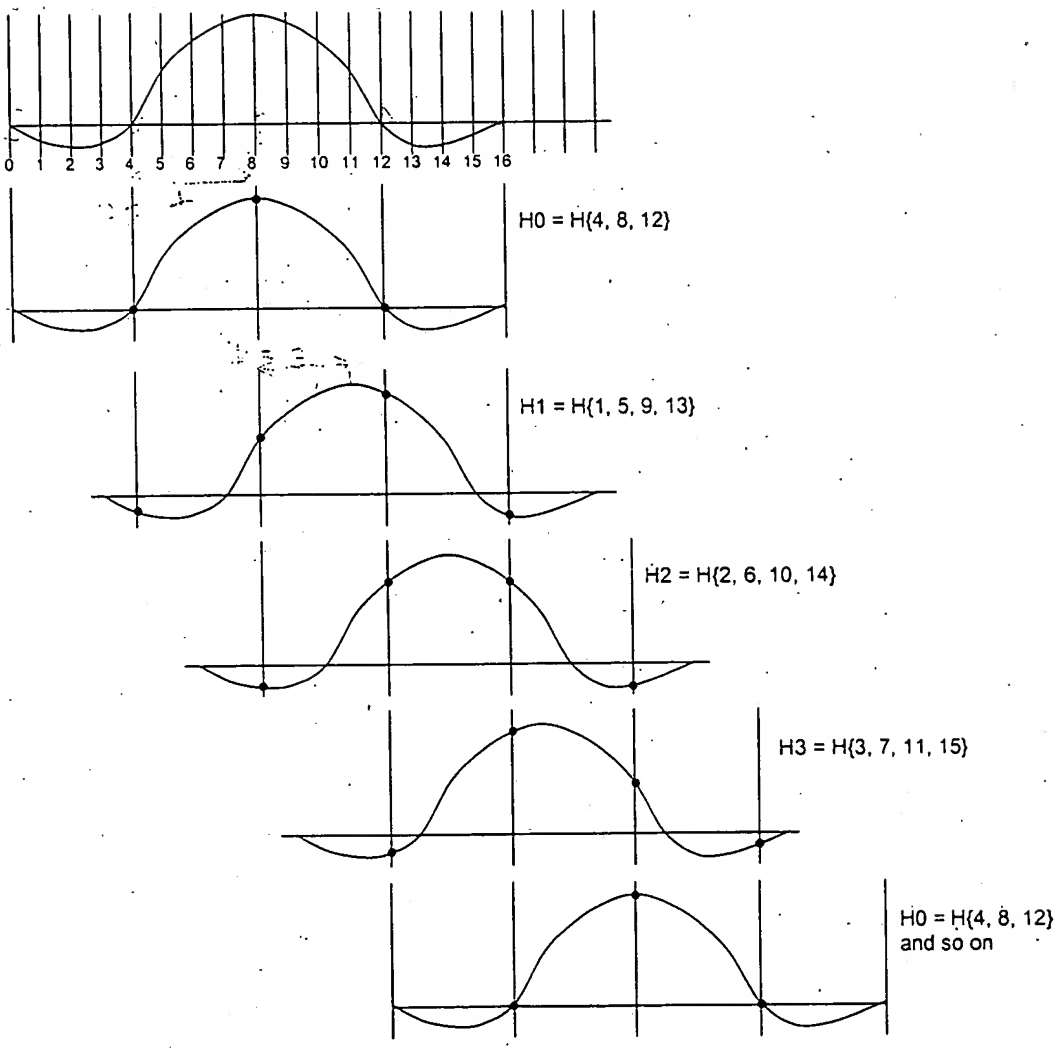

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
H0 = H{4, 8, 12}
H1 = H{1, 5, 9, 13}
H2 = H{2, 6, 10, 14}
H3 = H{3, 7, 11, 15}
H0 = H{4, 8, 12}
and so on

Fig. 10

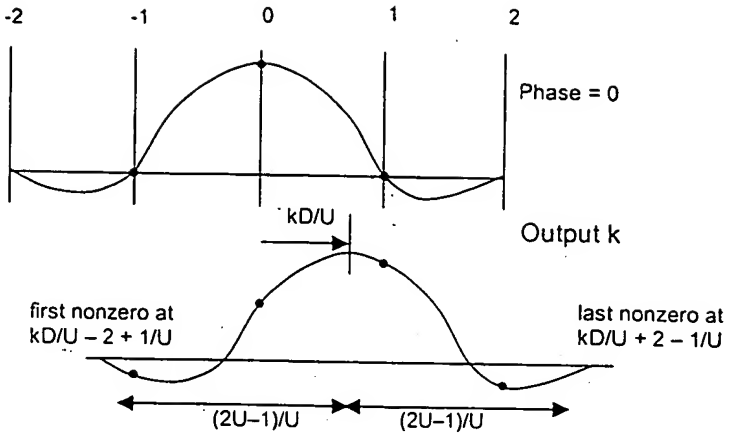


Fig. 11

Resampling factor	Multiply _factor	H_Arch _Kernel	filter taps	Data step to next group of outputs	Starting access data point
4/3 horizontal	1	1:1 slope 4 outs	4	n/a	-2
vertical			4	1	-2
5/3 horizontal	1	2:1 slope 4 outs	4	2	-1
vertical			5	1	-3
2/1 horizontal	2	2:1 slope 4 outs	4	n/a	-1
vertical			5	0	-1
7/3 horizontal	1	2:1 slope 4 outs	5	2	-2
vertical			6	0	-1
8/3 horizontal	1	4 tall	5	1	-1
vertical			6	0	-1
3/1 horizontal	1	4 tall	4	n/a	-1
vertical			4	0	-1
10/3 horizontal	1	4 tall	5	1	-1
vertical			6	0	-1

Fig. 13

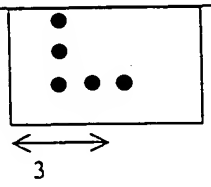
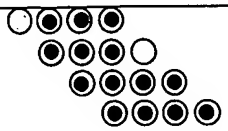
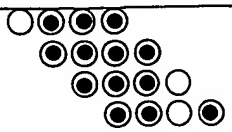
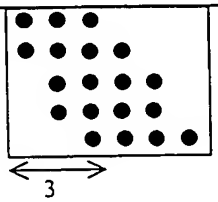
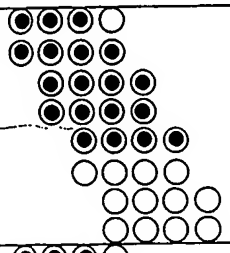
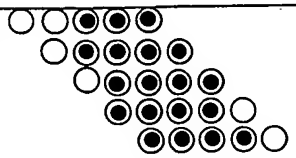
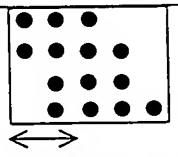
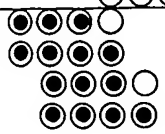
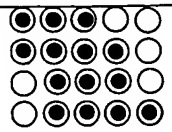
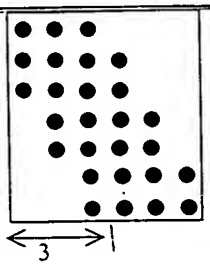
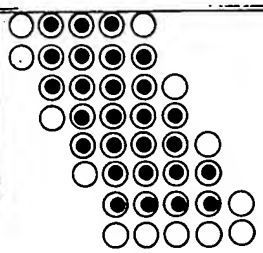
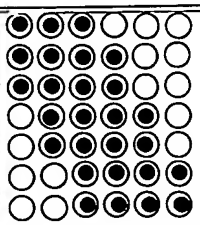
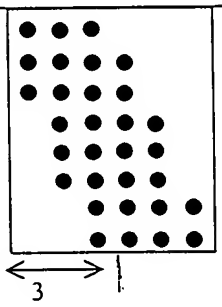
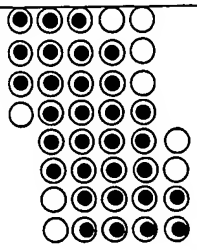
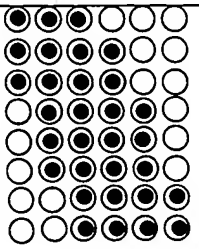
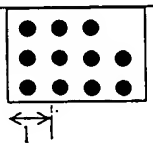
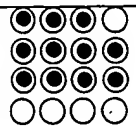
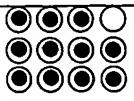
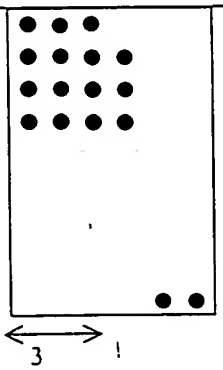
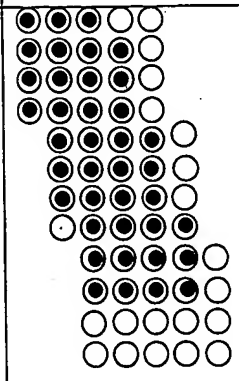
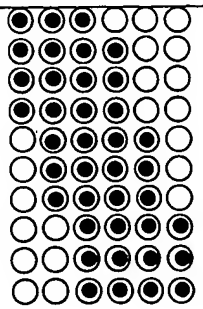
Resampling factor	Data access pattern	Horizontal pass coverage chart	Vertical pass-coverage chart
4/3			
5/3			
6/3 = 2			
7/3			
8/3			
9/3 = 3			
10/3			

Fig. 12